Haptic Collaboration over the Internet

João P. Hespanha, Margaret McLaughlin, Gaurav S. Sukhatme
Minoo Akbarian, Rajiv Garg, Weirong Zhu

Integrated Media Systems Center
University of Southern California
Los Angeles, CA 90089

Abstract

We address the real-time collection and simultaneous broadcast of haptic information to multiple haptic session participants, so that collaborative exploration of objects is possible, even when users are equipped with disparate haptic devices, such as the PHANToM and the CyberGrasp. We have designed and are currently testing a prototype system for haptic collaboration over the Internet. The basic idea is to provide a framework for multiple users (each with his or her own haptic device connected to a computer) to share a common experience of touch. This will allow users to exert forces on each other through the network as well as exert forces on common objects.

In this paper we present a distributed architecture for haptic collaboration via the Internet. We explicitly address the issue of latency (communication delay), thus providing a foundation for a shared haptic experience among distributed users. With respect to stability, latency is a critical factor that governs whether two users can truly share a common haptic experience. We propose an algorithm where the nature of the interaction between two hosts is decided dynamically based on the measured network latency between them. Users on hosts that are near each other (low communication latency) are dynamically added to fast local groups. If the communication latency is high, users are allowed a slower form of interaction where they can touch and feel objects but cannot exert forces on them. Users within a fast local group experience true haptic collaboration since the system is able to resolve the interaction forces between them fast enough to meet stability criteria. We discuss the creation, maintenance and update mechanisms of local groups for fast interaction, as well as synchronization mechanisms for hosts participating in slower interaction. We conclude with a discussion of open issues and future work.

1. Introduction

Haptic (adj): of or relating to the sense of touch. In the present context, haptic refers to the modality of touch and the sensation of shape and texture an observer feels when exploring an object in a virtual environment. Applications of haptics include online museums [6], aid for the visually impaired, remote surgery and entertainment. In many of these applications it will be necessary for users to **interact with each other** as well as with other objects. In this article, we propose an architecture for haptic collaboration among distributed users. We focus on collaboration over a non-dedicated channel (such as an Internet connection) where users experience stochastic, unbounded communication delays [7].

The area of haptic collaboration is relatively new. There have been a few prior studies that we briefly review here. In a study by Basdogan et. al. [1], partners at remote locations were assigned three cooperative tasks. Experiments were conducted with visual feedback only, and with both visual and haptic feedback. Both performance and feelings of togetherness were enhanced in the dual modality condition. Durlach and Slater [3] note that factors that contribute to a sense of co-presence include being able to observe the effect on the environment of actions by one's interlocutors, and being able to work collaboratively with co-present others to alter the environment. Buttolo et. al. [4] note that when the same virtual environment is shared between two distributed sites there may be registration problems. Representations of the virtual object must coincide, but the distributed nature of the communication, especially over the Internet, may introduce considerable latency whose effects may be hard to predict.

2. Virtual Haptic World

Imagine you decide to go to a handicraft museum. There is a map of the museum at the door showing different halls in the museum, each containing a group of handicrafts. Upon entry into a hall, you can see the handicrafts and the other people in that room. You can touch all of the objects in the room and interact with them. In a real museum, all of the above are familiar experiences, except for the last one. As a matter of practice, touching art objects is usually strictly prohibited.

The scenario described above motivates the research presented here. Our goal is to design an architecture that will support collaborative touch in virtual environments. We term such environment a ***virtual haptic world***. As shown in *Figure 1*, users may have different kinds of haptic devices, such as the PHANToM, CyberGrasp, or a FEELit mouse, or they can just be viewers. Some of the participants in the haptic world may only provide virtual objects as a service to the remaining users. This would be the role, e.g., of a museum's server.

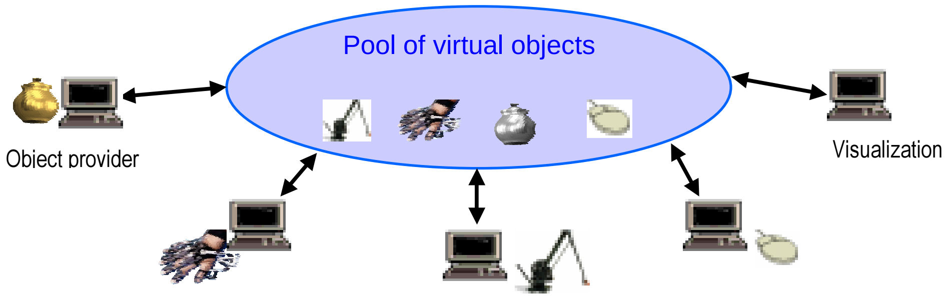


Figure 1: A virtual haptic world

From a computational perspective, a haptic world consists of a network of nodes. Each node corresponds to a computer whose operator is part of the shared virtual environment. The operator will typically interact with virtual objects through a haptic device, but conceivably, some users may interact with the haptic world using other modalities, e.g. by simple visualization. Some nodes may operate autonomously (i.e., without a human operator) and simply provide virtual objects for the haptic world.

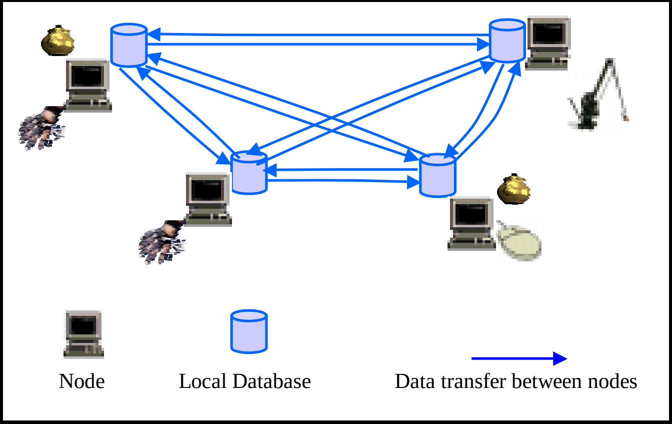
Each node in the haptic world contributes to the shared environment with virtual objects. These can be static, e.g., a sculpture "bolted" to the ground, or dynamic, e.g., a teapot that can be virtually manipulated. We view the haptic devices that the human operators use to interact with the haptic world as dynamic objects. Each object in the haptic world is *owned* by one of the nodes, which is responsible for defining how its dynamic properties evolve. Typically, a node that is physically connected to a haptic device owns the object that represents the device.

Two databases are used to represent a haptic world. The *node database* contains information about the node network. It stores the logical identifiers and the IP addresses of all nodes, as well as the latency and available bandwidth between all nodes. The need for this information will become clear later. This database is dynamic because new nodes may join or leave the haptic world at run-time. The *object database* contains the information about all objects that are part of the haptic world. Each record in this database refers to a particular object and it contains the object identifier, the identifier of the node that owns it, its static properties (shape, size, color, etc.) and its dynamic properties (position, orientation, velocity, etc.).

The force control algorithms used for haptic rendering generally require high sampling rates (typically, on the order of 1KHz) and low latency (typically, on the order of a few milliseconds) [5]. This means that the databases need to be queried very frequently and with very low delay. Because of this it is necessary to distribute these databases by keeping local copies at each node. This allows for very fast access to the data about the objects that is needed for the force feedback loops, at the expense of the added complexity introduced by issues related to the consistency between the databases. Much of what follows is precisely related to the problem of keeping the databases synchronized so that all nodes have roughly the same perspective on the shared environment.

3. Database Synchronization

Since the object database contains data that is dynamic, the local copies of this database that exist at each node must be kept synchronized by a periodic exchange of data. This is done by a very simple mechanism that uses the concept of object ownership introduced earlier: periodically, the owner of each object broadcasts the dynamic properties of its objects to all other nodes. Each node must then continuously listen to the other nodes for updates on the dynamic properties of the objects that it does not own. This is represented schematically in *Figure 2*.



```
While (1)
{
        Listen for dynamic properties of objects owned
            by other nodes and update local object
            database
        Query haptic rendering system for dynamic
            properties of objects owned
        Broadcast dynamic properties of objects owned
        Wait for one synchronization period
}
```

Figure 2: Object database synchronization in the haptic world.

Table 1: Pseudo-code for object database synchronization

Typically, the haptic rendering system uses the following fairly standard algorithm:

```
Compute amount of overlapping between objects owned and all other objects
Compute forces on objects owned (assuming spring-damper system)
Transmit forces to haptic device
Integrate forward in time to predict dynamic properties of objects owned at next sampling time
```

Table 2: Pseudo-code for the update of dynamic properties of objects

When the number of nodes is large, the broadcast of object properties required by the algorithm in *Table 1* may be costly unless the synchronization period is large. We will address this issue later.

Another main challenge arising from the distributed nature of the databases that store the information about the haptic world is related to the addition and removal of nodes from the haptic world. When a new node joins the haptic world, it must first receive the current node and object databases from some other node in the haptic world. It must then add itself to the node database and add its objects to the object database. Finally, it must inform all other nodes of these changes to the databases. This is implemented by the pseudo-code shown in *Table 3* that must run in every node.

```
Request copy of node database
Request copy of object database
Add self to node database
Add objects owned to object database
Broadcast request to add new record to node database
Broadcast request to add new records to object database
While (node active)
{
Listen for requests to:
    send node/object database
    add/remove record to/from node database
    add/remove record for/from object database
}
Broadcast request to remove self from node database
Broadcast request to remove owned objects from object
        database
```

```
Input: G = {list of objects in local group}

While (1)
{
    Listen for the dynamic properties of objects in G owned
        by other nodes and update local object database
    Query haptic rendering system for dynamic properties
        of objects owned
    Broadcast to the owners of the objects in G the dynamic
        properties of objects in G owned by self
    Wait for one local group synchronization period.
}
```

Table 3: Pseudo-code for the creation of a new node in the haptic world

Table 4: Pseudo-code for local group synchronization

4. Local Groups

The broadcast required by the synchronization algorithm in *Table 1* can be very costly when the number of nodes is large. Because of this, the synchronization period may need to be fairly long. For static objects this poses no problems, but the same is not true for dynamic objects, i.e., objects that can move.

When two or more dynamic objects touch each other, the resulting motion must be computed by simulating Newton's laws using an algorithm similar to the one in *Table 2*. However, when the same node does not own all the objects involved in a close interaction, each object only observes the effect of its motion in the motion of other objects at a relatively low sampling rate, determined by the synchronization period. This leads to very unrealistic motions (and possibly instability) because the algorithm in *Table 2* no longer provides a good approximation to Newton's law. We overcome this by creating small groups of nodes that engage in very fast and very frequent exchange of synchronization data for objects in close interaction. The creation of these groups is, of course, only possible when the bandwidth between the nodes is sufficiently large and the latency is sufficiently small. Because of the high cost of local groups, these should only be maintained while the objects are interacting.

As explained above, to resolve the motion of objects involved in close interaction a high bandwidth/low latency synchronization mechanism is needed. In our architecture this is achieved by introducing the concept of a *local group*. A local group consists of a group LG of objects, whose owners enhance the basic synchronization algorithm for those objects in LG, by decreasing the synchronization-sampling period. The local group synchronization algorithm, given in *Table 4*, is very similar to the basic one in *Table 1*.

Since each local group determines the positions of all the objects in that local group, each object should belong to, at most, one local group (this does not prevent a node that owns several objects from being involved in several local groups). Moreover, the fast synchronization within the local group requires high bandwidth and low latency between the nodes involved. Special care must therefore be paid to the creation of local groups.

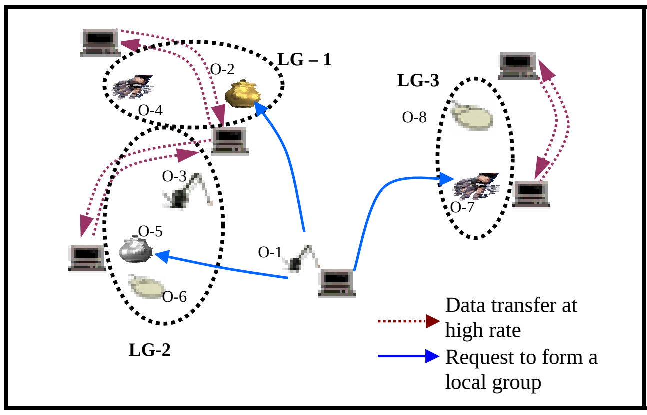


Figure 3: Haptic world with three local groups and a node requesting to create a local group

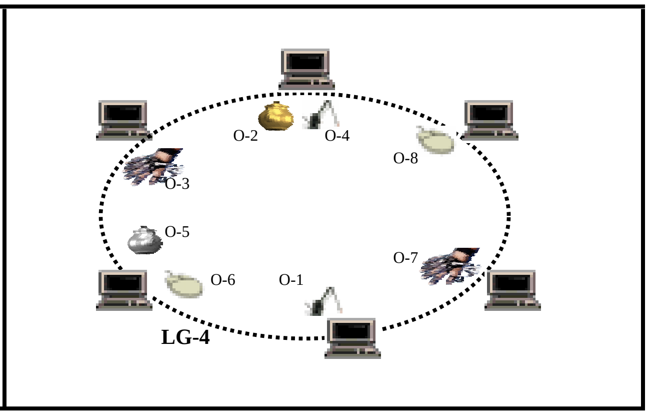


Figure 4: New local group, after the request in Figure 3 was processed

We use an example to illustrate the issues involved in the management of local groups. Consider the haptic world shown in *Figure 3*. In this figure we see three local groups: LG-1 is formed by the set of objects {O-2, O-4}, LG-2 is formed by {O-3, O-5, O-6}, and LG-3 is formed by {O-7, O-8}. Note that the same node owns the objects O-2 and O-3 but they are part of distinct local groups. This means that, although belonging to the same node, these objects are not in close proximity and therefore their motions are independent. Suppose now that the user at the node that owns O-1 wants to use O-1 to manipulate the objects O-2, O-5, and O-7 (*Figure 3*). This requires the creation of a local group that contains *T* = {O-1, O-2, O-5, O-7}. However, since some of these objects are already part of other local groups, the old local groups LG-1, LG-2, LG-3 must be destroyed and a new local group LG-4 must be created, containing the objects in *T* as well as those in the old local groups LG-1, LG-2, and LG-3 (*Figure 4*). This only occurs if the network connections between all the nodes that own the objects in question have sufficiently large bandwidth and sufficiently low latencies for the local group synchronization.

The pseudo-code in *Table 5* implements the algorithm used to create a new local group. The pseudo-code in *Table 3* also needs to be modified as shown in *Table 6* to process the requests generated by the algorithm in *Table 5*.

```
Input: T = {desired list of objects in new local group}

L = Expand (T)      % determine list of all objects that need
                    %  to be included in new local group
If Feasible (L)
{    % Only create local group if all nodes involved
     % satisfy the bandwidth and latency requirements
     For each l ∈ L
          Request owner of object to destroy the local group
               to which it belongs
     For each l ∈ L
          Request owner of object to create a local group for
               objects in L
     Return Success
}
Else
     Return Failure
```

Table 5: Pseudo-code to create a new local group

```
{...}

While (node active)
{
   Listen for requests to:
        send node database
        send object database
        add/remove record to node database
        add/remove record for object database
        create/destroy a local group
}

{...}
```

Table 6: Modification in the pseudo-code in Table 3 to process the requests generated by Table 5.

5. Conclusions and Future Work

We proposed an architecture for the real-time collection and simultaneous broadcast of haptic information to multiple haptic session participants, so that collaborative exploration of objects is possible, even when users are distributed across a network. The architecture relies on two distributed databases: the node and the object databases. These two databases are dynamic and need to be kept coherent among all nodes in the virtual haptic world. We presented pseudo-code for the algorithms that keep these databases synchronized. These algorithms are independent of the actual haptic devices employed by each user.

In future work, we hope to make significant progress on the registration of the haptic display systems in collaborative-networked environments. We will also examine the necessary entities to achieve networked collaboration with disparate haptic devices (pen-based versus glove-based, small versus large workspace). We plan to address not only integration issues but also questions related to the interaction process itself, including feelings of co-presence and performance satisfaction, and how these variables are affected by the exploration modality (vision, vision plus haptic or haptic only). Another line of research is the development of force control algorithms tailored to a distributed haptic environment. These algorithms must be robust with respect to the stochastic delays caused by the communication network.

6. References

[1] C. Basdogan, C. Ho, M. Slater, and M. A. Srinivasan, “The Role of Haptic Communication in Shared Virtual Environments,” PHANToM Users Group, 1998 (http://www.sensable.com/community/PUG98/19_basdogan.pdf).

[2] C. Ho , C. Basdogan, M. Slater, N. Durlach, M. A. Srinivasan , “An Experiment on the Influence of Haptic Communication on the Sense of Being Together,” 1998 (http://www.cs.ucl.ac.uk/staff/m.slater/BTWorkshop/touchexp.html).

[3] N. Durlach and M. Slater, “Presence in Shared Virtual Environments and Virtual Togetherness,” 1998 (http://www.cs.ucl.ac.uk/staff/m.slater/BTWorkshop/durlach.html).

[4] Buttolo, P., J. Hewitt, R. Oboe, and B. Hannaford, “Force Feedback in Virtual and Shared Environments,” 1997 (http://brl.ee.washington.edu/BRL/publications/Rep100.ps).

[5] Wilson, J. P, Kline-Schoder, R. J., Kenton, M. A., and Hogan, N, “Algorithms for Network-Based Force Feedback,” in Salisbury, J. K. and Srinivasan, M. A. (Eds), Proc. PHANTOM Users Group Workshop, MIT 1999.

[6] McLaughlin, M. L., Sukhatme, G., Hespanha, J., Shahabi, C., Ortega, A., & Medioni, G “The Haptic Museum,” Proc. EVA 2000, Conference on Electronic Imaging and the Visual Arts, 2000.

[7] McLaughlin, M. L., G. Sukhatme, J. Hespanha, C. Shahabi, and A. Ortega, “Touch in Immersive Environments,” Proc. EVA 2000 Conference on Electronic Imaging and the Visual Arts, Edinburgh, Scotland, 2000.